503 & 505 Software Engineering

Unit-1 Introduction, Software Requirement Analysis & Specification PART - 1

Prepared By: ARPIT PAREKH
SSCCS, Bhavnagar

Contents:

❖ Introduction to Software

The Introduction starts with the most common question regarding the ***"SOFTWARE"***. The question raises in our mind is that ***"What is Software & how it affects to everyone in the I.T. world or how it used to the people to fulfillment their need?"*** The answer is as simple as the question. The software is both product and a vehicle for delivering the information to user. The software is used by the virtually everyone in the society. Software is developed or engineered not manufactured Software engineer have the moral duty or obligation to build the reliable software that should not harm to other people. The software users are only concern with whether or not software products meet their expectations to make their tasks easier to complete. There are different types of approaches regarding Software but among them following definitions defines relevant word.

- **IEEE defines**
  "SOFTWARE as the collection of the computer programs, procedures, rules and associated document and data"
- **Alan Turing defines**
  "SOFTWARE is all information processed by computer system, programs and data"

- ***Note:*** - The first theory about the software was proposed by the Alan Turing in 1935

❖ Define Software Engineering

- **Definition**
  "***Software Engineering*** *is defined as the systematic approach to the development, operations, maintenance and retirement of software.*"

❖ Nature of Software:

- **Introduction:**

The nature of the software defines the characteristic of the software that include following some phase that defines the software and its nature. Depends on the nature of the software that anyone can get the idea how the software will work on the system. Let's see the characteristic of the software as the nature of the software.

- **Ease of Change:**

The first software 's characteristic of the software or called nature of the software is the Ease of change. The ease of change defines the easiness of the software that can identify as How software can be changed accordingly the client's requirement. The software industry may have the pressure to change the software even deployment so here we can get the idea about the nature of the software that the software can built as the dynamic mode. Change is the primary requirement for the Software Development Industry. So, analyst need to focus on the Ease of changes related to client's demand.

- **Rapid Evaluation of Technology:**

As the time change the software need to change so rapid evaluation need to focus by the software analyst. Make the creative changes on the software day by day as the time passed. Release new version of the software with identity modification in the software. The industry needs to train the software analyst as per the new trend of the software in that the old software helped to update the new software and make creative.

- **Very Low manufacturing cost:**

The last nature of the software that defines the manufacturing cost. The cost almost affects the client so before communicate about the software development the analyst and industry need to communicate about the manufacturing cost. The manufacturing cost depends on the software characteristic if the software is large in the developing the cost will be as per the time and if the software development is small scale in the nature the cost will define as per it. So here as per the time of the software development the manufacturing cost will be define but analyst need to be carefully that low-cost software will always affect the client.

Conclusion

The nature of the software determines the futuristic existence of the software. It defines whether your software has ability to survive against all other software or not…?

❖ Software Engineering Approach (Work)

Introduction:

The software analyst is the main part between the client and the industrial development. So, the software engineer has some duty to complete the requirement within the time limit with quality of software. Following approaches are considered as the work of the software analyst.

Phased Development Process

The approach of the analyst should to start with the phase development process that includes the main focus on the requirement of the client by using the communication. The main process to define the phased development is that can break the problem of developing the software into successfully performing the requirements. Here we briefly discuss these basic phases; each one of them will be discussed in more detailed in the following types. The following approaches software engineer need to be applied at the time of the software development

1. **Requirement Analysis**

   Requirement analysis is done in order to understand the problem of software system. The goal of the requirement analysis is to document the requirements in the software requirements specification documents (SRS). The analyst needs to communicate about the client's need and well understood the proper flow before starting the developing the project. The analyst should also need to identify the system need whether the system of the industry can achieve the requirement in proper time or not?

2. **Software Design**

   The second approach of software design is to plan the solution of problem specified. In other words, the analyst needs to focus on the following questions.

   - **What is needed?**
   - **How to satisfy?**
   - **How to develop/design?**

   Above questions support the answer of the detailed design which supports the interaction between the client and system. It also includes the high-level complex design as well as detailed design.

3. **Coding**

   Once design is completed most of the major decision about the system has been made. The coding phase affects both testing and maintenance efforts because the testing and maintenance cost much higher than coding cost. The main goal of coding is to reduce the testing and maintenance effort.

4. **Testing**

   Major quality control is used during the software development at the time after proper testing. Basic function of testing is to detect the defects. The testing fundamental uncover the requirement, design and coding error that can analyst can fix.

Managing Process

The Development phase is imported to develop the software in the linear way. The managing process defines the importance of the development process that managed by the analyst. The analyst needs to be careful that every process should be in the time. Only well-planned development process can develop well structure software within time limit. Managing the process required the information upon which the management decision.

❖ Software Process and It's Characteristics

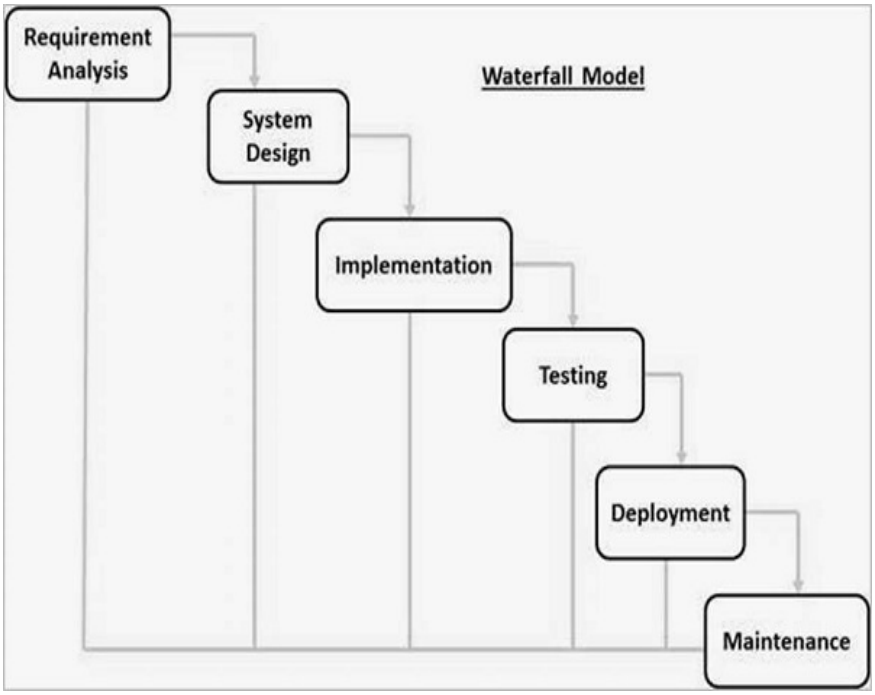
- **Introduction**
  As a process it may be used some desired characteristic for project development. These characteristics are important for working software process smoothly. It will help to understand basic requirements before starting process model. Following characteristics are necessary for development process.
- **Predictability**
  Predictability of a process determines how accurately the outcome of the following that process can be predicted before the project completed. Predictability can be considered as fundamental property of any process. In fact, if any process is not predictable, it is of limited use. If it was not predictable then there is no guarantee that doing a similar project using the process will secure for similar cost. A predictable process is also said to be under *statistical control*.
- **Testability and Maintainability**
  The testability and maintainability are the necessary for reducing the project cost. First, we define the concept of testing fundamental which clearly describe the flow of successful evaluation regarding the project and its process it is the necessary concept before deploying the project. After that mentality define the concept of maintain the project after deploying to the client. Maintainability describes the changes with updating of project.
- **Support Change**
  Support of changes defines how your project will change after deployment and a time limit. Changes are necessary for any deployed project because if changes will not define the project will not work for long time. Here we focus on the changes due to requirements will change after a time. As organization and business change the software support for business has change. In short changes are necessary for software updating.
- **Early Defect Removal**
  If there is an error in the requirements, then the design and the code will be affected by it. To correct the error in the requirements, then the design and the code will be affected by it. To correct the error after the coding is done would require both the design and the code to be changed. Error detection and correction should be a continuous process that is done throughout software development.
- **Process Improvement and Feedback**
  As earlier mentioned, a process is not static entity. To improve the quality is reducing the cost. Process improvements and feedback are necessary for maintain project quality after deployment and taking feedback we should define the quality for future enhancement. It is last concept of software process.

❖ Software Development Process Model

The process that deals with the technical and management issues of the software development is called "Software development process". In the software development process, we focus on activities directly related to production of the software unit. In this we discussed about module design, coding and testing. The management process is decided on the development process. Due to the importance of the process various models have been proposed. The following types of various models are going to discussed in this concept. There are various *software development life cycle models* defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models. Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

1) Waterfall Model

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a ***linear-sequential life cycle model***. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. The model was originally proposed by Royce.



- **Requirement Gathering and analysis** – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** – With the inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** – There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

Application of Waterfall Model

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate which are described as below.

- ✓ Requirements are very well documented, clear and fixed.
- ✓ Product definition is stable.
- ✓ Technology is understood and is not dynamic.
- ✓ There are no ambiguous requirements.
- ✓ Ample resources with required expertise are available to support the product.
- ✓ The project is short.

☺ Advantages

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one. Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order. Some of the major advantages of the Waterfall Model are as follows.

- ✓ Simple and easy to understand and use
- ✓ Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.

- ✓ Phases are processed and completed one at a time.
- ✓ Works well for smaller projects where requirements are very well understood.
- ✓ Clearly defined stages.
- ✓ Well understood milestones.
- ✓ Easy to arrange tasks.
- ✓ Process and results are well documented.

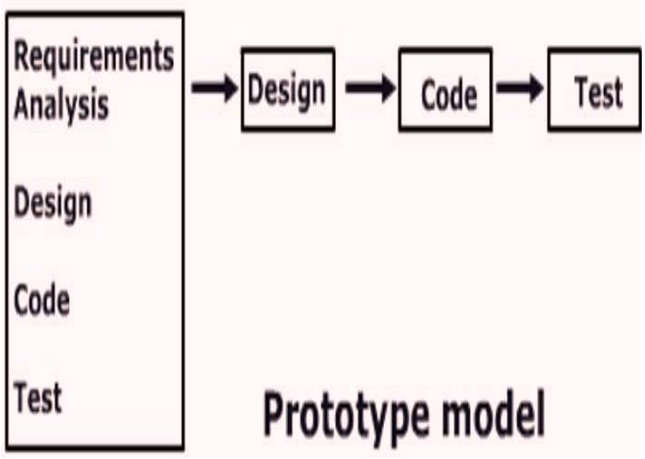
☹ **Disadvantages**

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage. The major disadvantages of the Waterfall Model are as follows.

- ✓ No working software is produced until late during the life cycle.
- ✓ High amounts of risk and uncertainty.
- ✓ Not a good model for complex and object-oriented projects.
- ✓ Poor model for long and ongoing projects.
- ✓ Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- ✓ It is difficult to measure progress within stages.
- ✓ Cannot accommodate changing requirements.
- ✓ Adjusting scope during the life cycle can end a project.

2) Prototyping Model (Throw-away prototype model)

The Software Prototyping refers to building software application prototypes which displays the functionality of the product under development, but may not actually hold the exact logic of the original software. Software prototyping is becoming very popular as a software development model, as it enables to understand customer requirements at an early stage of development. It helps get valuable feedback from the customer and helps software designers and developers understand about what exactly is expected from the product under development. A Software Prototype can be defined as a partial implementation of a system whose purpose is to learn something about the problem being solved or the solution approach. Prototyping is the rapid development of a system. Prototyping is a method that can be used for complex problem analysis. It is being very popular. It will encounter the first two limitation of the waterfall model.



The idea behind prototyping is that clients and the users can access their needs much better if they can see the working of a system, even if the system is only partial system. Prototyping emphasizes that actual practical experience is the best aid for understanding needs. Experience is gained by putting the system to use with actual client and end users using it. Based on their feedback; the prototype is modified to incorporate feasible changes of the suggested changes and the clients are again allowed to use the system. This cycle repeats until the benefit from further modification in the system and obtaining feedback is outweighed by the cost and time involved n making the changes and obtaining feedback the feedback.

When to Use:

- ✓ When it is very difficult to obtain exact requirement from custom
- ✓ The model is developed based on feedback so, feedback is necessary

- ✓ Complete built sample model is shown to user and based on his feedback SRS/Documents are prepaid.

☺ **Advantages**

- ✓ This type of model specially used for non-IT which is not good at specifying their requirement.
- ✓ When analyst is not confident about developer's capability, he asks for small prototype to built.
- ✓ It reduces risks of failure as potential risks identify early.
- ✓ Missing services may be detected and confusing services may be identified
- ✓ A working system is available early in the process
- ✓ Improved system usability, design quality and maintainability.
- ✓ Reduced overall development effort

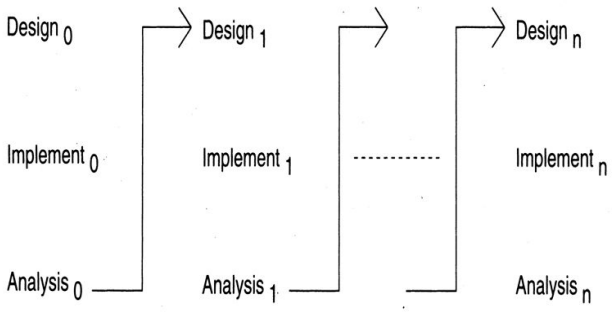
☹ **Disadvantages**

- ✓ Making prototype is a slow process in contrary to another model.
- ✓ Too many changes can be disturbing the rhythm of the development team.
- ✓ Some Rapid change may occur to uselessness of the whole model. So, sometimes it is called as ***Throw-away prototype model.***

3) Iterative Enhancement Model (Incremental model)

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. Each iteration design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).The following illustration is a representation of the Iterative and Incremental model. Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. During the software development process more than an iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach. In this incremental model, the whole requirement is divided into various builds. At the time of iteration, the development module goes through the requirements, design, implementations and testing phases. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is ready as per the requirement. The key to a successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification & testing of each version of the software against those requirements within each cycle of the model. As the software evolves through successive cycles, tests must be repeated and extended to verify each version of the software.

Typical Flow of Iterative Enhancement



Iterative Model - Application

Like other SDLC models, Iterative and incremental development has some specific applications in the software industry. This model is most often used in the following scenarios

- ✓ Requirements of the complete system are clearly defined and understood.

- ✓ Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- ✓ There is a time to the market constraint.
- ✓ A new technology is being used and is being learnt by the development team while working on the project.
- ✓ Resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.
- ✓ There are some high-risk features and goals which may change in the future.

☺ **Advantages**

The advantage of this model is that there is a working model of the system at a very early stage of development, which makes it easier to find functional or design flaws. Finding issues at an early stage of development enables to take corrective measures in a limited budget. The advantages of the Iterative and Incremental SDLC Model are as follows

- ✓ Some working functionality can be developed quickly and early in the life cycle.
- ✓ Results are obtained early and periodically.
- ✓ Parallel development can be planned.
- ✓ Progress can be measured.
- ✓ Less costly to change the scope/requirements.
- ✓ Testing and debugging during smaller iteration is easy.
- ✓ Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
- ✓ Easier to manage risk - High risk part is done first.
- ✓ With every increment, operational product is delivered.
- ✓ Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
- ✓ Risk analysis is better.
- ✓ It supports changing requirements.
- ✓ Initial Operating time is less.
- ✓ Better suited for large and mission-critical projects.
- ✓ During the life cycle, software is produced early which facilitates customer evaluation and feedback.

☹ **Disadvantages**

The disadvantage with this SDLC model is that it is applicable only to large and bulky software development projects. This is because it is hard to break a small software system into further small serviceable increments/modules. The disadvantages of the Iterative and Incremental SDLC Model are as follows

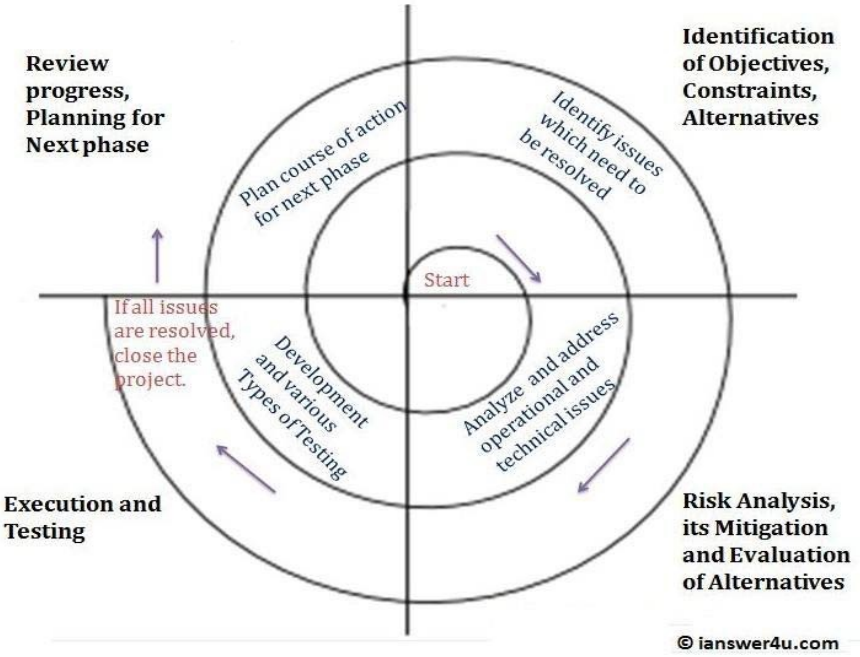
- ✓ More resources may be required.
- ✓ Although cost of change is lesser, but it is not very suitable for changing requirements.
- ✓ More management attention is required.
- ✓ System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- ✓ Defining increments may require definition of the complete system.
- ✓ Not suitable for smaller projects.
- ✓ Management complexity is more.
- ✓ End of project may not be known which a risk is.
- ✓ Highly skilled resources are required for risk analysis.
- ✓ Project's progress is highly dependent upon the risk analysis phase.

4) Spiral Model

Introduction

The spiral model originally proposed by **Boehm [BOE88].** In the spiral model software is developed in a series of incremental releases. The spiral model is divided into number of framework activities, also called task regions. For small projects, the number of work tasks & their formality is low. For large & critical Project, the

number of work tasks & their formality is high. As this evolutionary process begins, the software engineering teams move around the spiral in a clockwise direction, beginning at the core.



The first circuit around the spiral might result in the development of a product specification. Subsequent passes around the spiral might be used to develop prototype and then progressively more sophisticated version of the software. Each pass through the planning region results in adjustment to project plan. Cost & Schedule are adjusted based on feedback derived from customer evaluation. The Project manager decides the number of iterations required to complete software.

Each cube placed along the axis represents the starting points of different types of project. At the core the Concept Development Project start, and work until concept is completed. If we process for new product then proceed next cube and that cube consider as New Product Development Projects. The spiral model remains operative until the software is retired. There are time when the process is dormant,but whenever a change is initiated,starts the process at appropriate points,e.g. Production Enhancement Project.

☺ **Advantages**

- ✓ The spiral model is realistic approach to development the large scale system & software
- ✓ The spiral model uses prototyping as a risk reduction mechanism, but more important,apply prototyping approach at any stage in evolution of product.
- ✓ The spiral model demands a direct consideration of technical risks at all stages of the project.
- ✓ It is suitable for high risk projects, where business needs may be unstable.

☹ **Disadvantages**

- ✓ It is a complicated approach especially for projects with a clear SRS.
- ✓ If any major risk is uncovered at last definitely problem will occurs.
- ✓ It is not suitable for low risk projects.
- ✓ Skills required, evaluating and reviewing project from time to time, need expertise.

Difference between Waterfall model and Prototype model:

S.No.	WATERFALL MODEL	PROTOTYPE MODAL
1.	Waterfall model is a software development model and works in sequential method.	Prototype model is a software development model where a prototype is built, tested and then refined as per customer needs.

2.	It gives emphasis on risk analysis.	It does not give emphasis on risk analysis.
3.	There is high amount risk in waterfall model.	It is suitable for high-risk projects.
4.	In this, quick initial reviews are possible.	In this, quick initial reviews are not possible.
5.	It is best suited when the customer requirements are clear.	It is best suited when the requirement of the client is not clear and supposed to be changed.
6.	In this user Involvement is only at the beginning.	In this user involvement is high.
7.	It supports automatic code generation as. results in minimal code writing.	It does not support automatic code generation.
8.	The complexity of an error increases as the nature of the model each phase is sequential of the other.	The complexity of an error is low as the prototype enables the developer to detect any deficiency early at the process.
9.	Flexibility to change in waterfall model is Difficult.	Flexibility to change in prototype model is Easy.

MCQ

A step-in waterfall model that involves a meeting with the customer to understand the requirements.

a) Requirement Gathering

b) SRS

c) Implementation

d) Customer review

Methodology in which project management processes were step-by step.

a) Incremental

b) Waterfall

c) Spiral

d) Prototyping

Q2. Which model is best suitable when basic requirement are clear only some changes are made wit

a) Prototyping model

b) Spiral model

c) Evolutionary model

Who is first described spiral model?
Barry Boehm

Spiral model is a combination of which of two model?
Waterfall model and iterative model

Every phase in the Spiral model is **start with a _____ and ends with the _____.**
Every phase in the Spiral model is **start with a design goal and ends with the client review.**

[1] Spiral model originally proposed by
a. Boehm
b.Winston
c.Royce
d.Dexter

[2] In the spiral model 'risk analysis' is performed
a. In the first loop
b .In every loop
c.Before using the spiral model
d.in first and second loop

[3] The spiral model was originally proposed by
a.IBM
b.Pressman
c.Royce
d.Barry Boehm

[4] Identify the disadvantage of the Spiral Model.
a) Doesn't work well for smaller projects
b) High amount of risk analysis
c) Strong approval and documentation control
d) None of these

[5] Each loop in the spiral represents _____ of the software process in Boehm's spiral model
a. phase
b. design
c. documentation
d. none of the above

[6] Which of the following property does not correspond to a good Software Requirements Specification (SRS) ?
a) Verifiable
b) Ambiguous
c) Complete
d) Traceable

[7] Which of the following property of SRS is depicted by the statement: "Conformity to a standard is maintained"?
a) Correct
b) Complete
c) Consistent
d) Modifiable

[8] Which of the following statements about SRS is/are true?
i. SRS is written by customer
ii. SRS is written by a developer
iii. SRS serves as a contract between customer and developer
a) Only i is true
b) Both ii and iii are true
c) All are true
d) None of the mentioned

[9] The SRS document is also known as _____ specification.
a) black-box
b) white-box
c) grey-box
d) none of the mentioned

[10] Consider the following Statement: "The output of a program shall be given within 10 secs of event X 10% of the time." What characteristic of SRS is being depicted here?
a) Consistent
b) Verifiable
c) Non-verifiable
d) Correct

[11] Narrative essay is one of the best types of specification document?
a) True
b) False